

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр.8304

Бочаров Ф.Д.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Изучить основы рекурсии и составления эффективных алгоритмов.

Постановка задачи.

- 1) Разработать программу, использующую рекурсию;
- 2) Сопоставить рекурсивное и итеративное решение задачи;
- 3) Сделать вывод о целесообразности и эффективности рекурсивного подхода для решения данной задачи.

Вариант 12

Построение синтаксического анализатора для понятия *скобки*.

скобки ::= квадратные | круглые | фигурные

квадратные ::= [круглые фигурные] | +

круглые ::= (фигурные квадратные) | -

фигурные ::= {квадратные круглые} | 0

Описание алгоритма.

Для решения поставленной задачи был реализован набор взаиморекурсивных булевых функций, структурно совпадающих с определением анализируемого объекта из условия задачи. На каждом этапе программа анализирует данные и передает их в другую функцию, таким образом происходит полный анализ входных данных, на основании чего в самый внешний вызов возвращается результат — true или false.

Спецификация программы.

Программа предназначена для синтаксического анализа выражения методом рекурсии.

Программа написана на языке C++. Входными данными является либо строка, либо путь до файла содержащего строку (строки). Выходными данными являются промежуточные значения входной после замены описанной выше и глубина рекурсии.

Описание функций.

Функция `bool isbracketsequence(string);` - Основная функция, реализующая описание определения.

`bool issquare(string); bool`

`iscircle(string); bool`

`isfigure(string);` Данные

функции `bool`

`issquare(string);`

Осуществляет проверку на то, удовлетворяет ли входная строка поставленной задаче. Функция принимает проверяемую строку, ссылку на поток вывода и имеет параметр `snt`, который по умолчанию равен 0, он используется для подсчета символов. Программа по условию проверяет на количество символов.

Циклом ищет '(' и закрывающую ')', проверяет на количество символов. Если условие выполняется, то записывает содержимое в подстроку и далее передает строку в другую функцию.

Вывод.

Был получен опыт работы с рекурсией и с построением синтаксического анализатора. На мой взгляд, итеративное решение поставленной задачи более эффективно.

ПРИЛОЖЕНИЕ

1) ТЕСТИРОВАНИЕ:

Работа программы для строки

(((0+){+-})({+-}[-0]))(({+-}[-0]){{[-0](0+)}}))

```
Analyzing issquare : <<[<0+><+->]<+->[-0]>>[<+->[-0]><[-0]<0+>>]>
Analyzing iscircle : <<[<0+><+->]<+->[-0]>>[<+->[-0]><[-0]<0+>>]>
Analyzing isfigure : <[<0+><+->]<+->[-0]>
Analyzing issquare : [<0+><+->]
Analyzing iscircle : <0+>
Analyzing isfigure : <+->
Analyzing iscircle : <<+->[-0]>
Analyzing isfigure : <+->
Analyzing issquare : [-0]
Analyzing issquare : [<<+->[-0]><[-0]<0+>>]
Analyzing iscircle : <<+->[-0]>
Analyzing isfigure : <+->
Analyzing issquare : [-0]
Analyzing isfigure : <[-0]<0+>>
Analyzing issquare : [-0]
Analyzing iscircle : <0+>
<<[<0+><+->]<+->[-0]>>[<+->[-0]><[-0]<0+>>]> == Its bracket
```

Таблица результатов ввода/вывода тестирования программы

Входная строка	Вывод
	программы
+	True
-	True
0	True
2	True
[(0+){+-}]	True
{{[-0](0+)}}	True
(({+-}[-0])	True
[(({+-}[-0]){{[-0](0+)}}]	True
(([-0](0+)){{0+}{+-}})	False
{[[0+]{+-}]{+-}[-0])}	False
(([(0+){+-}]{+-}[-0]))(({+-}[-0]){{[-0](0+)}})	True

2) ИСХОДНЫЙ КОД:

```
#include<vector>
#include<fstream>
#include<iostream>
#include<string>
#include<iostream>

bool isbracketsequence(std::string);
bool issquare(std::string);
bool iscircle(std::string);
bool isfigure(std::string);

bool isbracketsequence(std::string str)
{
    return issquare(str) || iscircle(str) || isfigure(str);
}

bool issquare(std::string str)
{
    if(str == "+")
        return true;

    if(str.size() < 4) return false; // По условию кв.ск. =
[ab], при этом не менее 4 символов

    int cnt = 0;

    for(int i = 1; i <= str.size() - 2; i++) // не учитываем
первую скобку и последнюю
    {
        if(str[i] == '(') // по условию круглые
            cnt++;
        if(str[i] == ')')
            cnt--;
        if(!cnt)
        {
            cnt =
            i; break;
        }
    }

    if(cnt == str.size() - 2)
        return false; // в строке нет места для других скобок

    std::string substr1 = "";
    std::string substr2 = "";
```

```

        for(int i = 1; i <= cnt; i++)
            substr1 += str[i]; // записываю первые
        скобки
        for(int i = cnt + 1; i <= str.size() - 2; i++)
            substr2 += str[i]; // вторые скобки

    std::cout << "    Analyzing issquare : " << str << std::endl;

    return ((str[0] == '[') && (str[str.size() - 1] == ']') &&
        iscircle(substr1) && isfigure(substr2));
}

bool iscircle(std::string str)
{
    if(str == "-")
        return true;

    if(str.size() < 4) return
    false;
    int cnt = 0;

    for(int i = 1; i <= str.size() - 2; i++)
    {
        if(str[i] == '{')
            cnt++;
        if(str[i] == '}')
            cnt--;
        if(!cnt)
        {
            cnt =
            i; break;
        }
    }

    if(cnt == str.size() -
        2) return false;

    std::string substr1 = "";
    std::string substr2 = "";

    for(int i = 1; i <= cnt; i++)
        substr1 += str[i];
    for(int i = cnt + 1; i <= str.size() - 2; i++)
        substr2 += str[i];

    std::cout << "    Analyzing iscircle : " << str <<

    std::endl;
    return
    ((str[0]

    == '(') && (str[str.size() - 1] == ')') &&
    isfigure(substr1) && issquare(substr2));
}

```

```

bool isfigure(std::string str) {

    if(str == "0")
        return true;

    if(str.size() < 4) return

    false; int cnt = 0;

    for(int i = 1; i <= str.size() - 2; i++)
    {
        if(str[i] == '[')
            cnt++;
        if(str[i] == ']')
            cnt--;
        if(!cnt)
        {
            cnt =
                i; break;
        }
    }

    if(cnt == str.size() -
        2) return false;

    std::string substr1 = "";
    std::string substr2 = "";

    for(int i = 1; i <= cnt; i++)
        substr1 += str[i];
    for(int i = cnt + 1; i <= str.size() - 2; i++)
        substr2 += str[i];

    std::cout << "    Analyzing isfigure: " << str <<

    std::endl; return ((str[0] == '{') && (str[str.size() - 1]

    == '}') &&
    issquare(substr1) && iscircle(substr2));
}

int main(int argc, char* argv[])
{
    int choice = 0; int
    k = 0; char tmp;
    std::string str;
    std::ifstream input;
    std::string proc_str
    = ""; while(true) {

```

```

std::cout<<"      What type of test do you want to do?"<<
std::endl;
std::cout<<"          1) from the file"<< std::endl;
std::cout<<"          2) from the console"<< std::endl;
std::cout<<"      enter any other number to exit."<<
std::endl;
std::cin >>
    choice;switch(cho
ice) {
case1:
    std::cout <<"* FROM THE FILE *"<< std::endl;
    input.open("../Tests/test.txt");
    if(!input)
    {
        std::cout <<"Couldnt open file"<<std::endl;
    }
    while(input.get(tmp)) {
        if(tmp !='\n')
        {
            str +=tmp;

        }

        else
        {
            std::cout <<"test "<<+k<<" for string - "<< str <<
            std::endl << std::endl;
            if(isbracketsequence(str))
            std::cout<<"          "<< str <<" == Its
            bracket"<< std::endl <<std::endl;
            else
            std::cout<<"          "<< str <<" == Its not
            bracket"<< std::endl <<std::endl;
            str ="";
        }
    }

    break;
case2:
    std::cout <<"* FROM THE CONSOLE *"<<
    std::endl;while(true) {
    std::cin >> str;
    if(isbracketsequence(str))
    std::cout<<"          "<< str <<" == Its
    bracket"<< std::endl <<std::endl;
    else
    std::cout<<"          "<< str <<" == Its not
    bracket"<< std::endl <<std::endl;
    str ="";

    break;
default:

```



```
        std::cout<<"                * EXIT *  
        "<<std::endl;  
        break;  
    }  
  
    }  
  
}
```